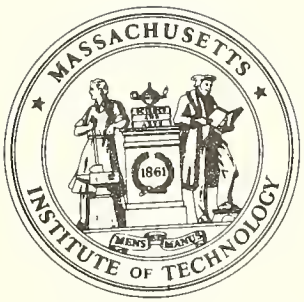

LIBRARY
OF THE
MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

Digitized by the Internet Archive
in 2011 with funding from
Boston Library Consortium Member Libraries

http://www.archive.org/details/matrixoperations00hall

working paper
department
of economics

Matrix Operations in Econometrics

by

R.E. Hall

Number 1 - July 20, 1967

MASS. INST. TECH.
JUN 18 1968
DEWEY LIBRARY

massachusetts
institute of
technology

50 memorial drive
cambridge, mass. 02139

Matrix Operations in Econometrics

by

R.E. Hall

MASS. INST. TECH.
JUN 18 1968
DEWEY LIBRARY

Number 1 - July 20, 1967

Econometricks Working Paper # 2

Dewey

MASS. INST. TECH.
OCT 22 1975
LIBRARIES

GENERAL BOOKBINDING CO.
75 360HY 13 061 BB
QUALITY CONTROL MARK
5539

R.E. Hall
July 20, 1967

MATRIX OPERATIONS IN ECONOMETRICS*

This paper presents a unified summary of the matrix operations which occur frequently in econometrics, and describes in detail the implementation of these operations on a digital computer.

We begin with a discussion of methods for storing matrices. This problem arises because of the essentially linear nature of a computer's memory which makes it necessary to store two-dimensional arrays as one-dimensional vectors. For general matrices (those with no restrictions on their elements, as distinguished from triangular, symmetric, or diagonal matrices), an obvious method is available for reduction to vector form: we simply stack the columns one after another. From the two-dimensional matrix A_{ij}, we generate the vector A^*_k, with i, j, and k related by the identity $k = (j - 1)\, m + i$, where m is the number of rows in A_{ij}. This is the method of matrix storage adopted by the authors of FORTRAN, and is now fairly standard.

Most operations in econometrics, however, involve triangular or symmetric matrices, for which the method of storage described for general matrices is wasteful of storage, since space is allocated to elements known to be zero (in the case of triangular matrices) or known to be equal to other elements (in the case of symmetric matrices). To avoid this waste, some form of compact storage is usually adopted for triangular matrices, and then only one triangle of symmetric matrices is stored. For example, the authors of IBM's Scientific Subroutine Package (2) use the following identity:

$$k = \frac{(j - 1)j}{2} + i$$

---

*This is a preliminary version of an appendix to R.E. Hall, Single Equation Methods in Econometrics.

532206

While this method uses storage perfectly efficiently, it does so at a large cost in computational efficiency. Evaluation of the SSP formula requires four arithmetic operations each time an element of A_{ij} is referenced. More importantly, there is no simple way to step through the elements of a particular row of a triangular matrix in sequence. In the case of a general matrix, the elements of a row are evenly spaced at intervals of m elements. Computationally efficient routines for operations on general matrices take advantage of this property to obtain a speed advantage of at least 100 per cent over routines which recompute k from i and j at each step.

We will now describe a storage method which retains the computational advantage of the general storage method while approaching the storage efficiency of the compact triangular method. This new method is based on the observation that the gaps which are left if an upper triangular matrix is stored as a general matrix are exactly the right size to hold a lower triangular matrix of the same size. That is, $(n + 1)n$ elements of storage will exactly accommodate a lower triangular matrix of order n starting at the first element and an upper triangular matrix of order n starting at the n+1st element. If upper and lower triangular matrices always appear in pairs this scheme will be perfectly efficient in its use of storage.

In econometrics, it appears that if the convention is adopted of storing triangular matrices as upper triangles and of storing only the lower triangles of symmetric matrices, fairly high storage efficiency can be attained, since frequently these two kinds of matrices appear in pairs. This convention is assumed in the following discussion.

A surprisingly large fraction of the matrix operations required in econometrics can be described in terms of repeated applications of the elementary operation of forming an inner product [the inner product, p, of two vectors of length n, x_i and y_i is defined as

$$p = \sum_{i=1}^{n} x_i y_i$$

and is often written $p = x'y$ or $p = (x,y)$].This is, of course, obviously true for matrix multiplication; it is less well-known that it is true for matrix inversion.

The knowledge that inner product calculations are at the heart of almost all of the operations can be used to great advantage in writing programs to implement the operations. In the system to be described, a basic inner product routine has been carefully coded in the assembly languages of the various machines it is intended to be used on; all the other routines are written in a form of FORTRAN acceptable to all of the machines. The resulting system is only a few per cent slower than a system written fully in assembly language and yet it involved only small fraction of the programming effort. The only serious drawback of this method is the relatively large amount of storage consumed by the kind of code generated by FORTRAN compilers.

In naming the routines of this system, we have adopted the following method: the first one, two, or three letters of the name give the matrix type (G for general, T for upper triangular, and Y for symmetric stored as a lower triangle); the last few letters are an abbreviation of the name of the operation (MLT for multiplication, INV for inversion, CVAL for characteristic values and FACT for factorization). Further, if the digit 2 appears after a Y,G, or T, it indicates that the same matrix is involved twice, as for example in T2YMLT which forms the symmetric matrix TT' from a triangular matrix T.

We will now describe the algorithms, computational methods, and calling sequences for the routines which were in existence at the time of this writing.

1. INPROD

This routine calculates the inner product $\sum_{i=1}^{n} x_i y_i$ of the two vectors x and y. Provision is made for allowing x or y to be a row of a matrix. All versions use extended precision in accumulating the inner product, truncating to regular length before returning (see Wilkinson, [3]).

Calling sequence: CALL INPROD(N,JSA, JSB,A,B,P).

N Number of elements entering inner product.

JSA Skip parameter for vector A. Every JSAth element is used; thus if JSA is 1, A could be a column, while if JSA is m, A could be a row of a matrix with m rows.

JSB Skip parameter for vector B.

A First vector.

B Second vector.

P Inner product.

Most FORTRAN compilers permit subvectors of larger vectors to be defined implicitly by using a subscript in the CALL statement (The unfortunate exception to this is the MADTRAN-MAD compiler on the MIT CTSS system). For example, if we wished to calculate the inner product of the third row of the 4 X 4 matrix C and the second column of the 4 X 4 matrix D, we would write

CALL INPROD[4,4,1,C(3),D(5),P] .

The user should study this example until he is sure he understands it before using this routine.

2. GGGMLT

This routine calculates the general matrix C as the matrix product AB of the two general matrices A and B. The matrix product is defined as

$$C_{ij} = \sum_{k=1}^{q} A_{ik} B_{kj} \quad ;$$

that is, the element in the ith row and jth column of C is the inner product of the ith row of A and the jth column of B.

Calling sequence: CALL GGGMLT(IFTRAN,NROWA,NCOLB,N,A,B,C)

IFTRAN If this parameter is zero, the regular product C = AB is calculated. If it is one, A is (logically) transposed before the multiplication, and C = A'B is calculated. A and B remain unchanged in either case.

NROWA Number of rows in matrix A. If IFTRAN is 1, this is the number of rows after transposition.

NCOLB Number of columns in B.

N Number of columns in A. Also must be equal to the number of rows in B.

A Input matrix with NROWA rows and N columns.

B Input matrix with N rows and NCOLB columns.

C Output matrix with NROWA rows and NCOLB columns.

Note that vectors are perfectly good general matrices with either one row or one column; no inefficiency results from using GGGMLT on vectors. Note also that the matrix C must be distinct from A and B, although A and B may be the same matrix.

3. G2YMLT

This routine forms the symmetric matrix X'X from the general matrix X; its main use in econometrics is to form the matrix of sums of cross-products.

Calling sequence: CALL G2YMLT(NROWX,NCOLX,X,Y)

NROWX	Number of rows in X.
NCOLX	Number of columns in X.
X	Input matrix.
Y	Output matrix, stored in lower triangle only. Upper triangle is left unchanged. Y has NCOLX rows and NCOLX columns.

4. GTGMLT

This routine forms the general matrix C as the product AB of the general matrix A and the upper triangular matrix B. It differs from GGGMLT only in that the elements of B below the diagonal do not enter the calculation.

Calling sequence: CALL GTGMLT(NROWA,N,A,B,C)

NROWA	Number of rows in A.
N	Number of columns in A. Also equal to the number of rows and columns of B.
A	General matrix with NROWA rows and N columns.
B	Triangular input matrix.
C	Output matrix with NROWA rows and N columns.

The columns of C are formed in reverse order so that A and C may be the same matrix. The main use of this routine in econometrics is in applying an orthonormalizing transform to a matrix of raw data.

5 . T2YMLT

This routine forms the symmetric matrix C = AA' from the upper triangular matrix A.

Calling sequence: CALL T2YMLT(N,A,C)

N	Number of rows and columns in A and C.
A	Upper triangular input matrix.
C	Lower triangular output matrix.

A and C may interlace each other in the manner described in the introduction; for example, if D has $N^*(N + 1)$ locations and an upper triangular matrix has its first element at D(N+1),

CALL T2YMLT[N,D(N+1),D(1)]

will form the symmetric product matrix correctly, starting at the first element of D.

6. TGGMLT

This routine forms the general matrix C as the product C = AB or alternatively C = BA', where A is an upper triangular matrix and B is a general matrix.

Calling sequence: CALL TGGMLT(IFTRAN,N,NB,A,B,C)

IFTRAN	If IFTRAN is 0, C = AB is formed. If IFTRAN is 1, C = BA' is formed.
N	Number of rows and columns in A. Also equal to number of rows in B if IFTRAN is 0, or number of columns in B if IFTRAN is 1.
NB	If IFTRAN is 0, NB is the number of columns in B. If IFTRAN is 1, NB is the number of rows in B.
A	Upper triangular input matrix.
B	General input matrix.
C	General output matrix, same number of rows and columns as B.

B and C may be the same matrix.

7. VGVMLT

This routine calculates the vector C as the product of the vector A and the general matrix B. It is intended to be used in cases where GGGMLT is inappropriate because the elements of A are not adjacent; it therefore includes a skip parameter for A.

Calling sequence: CALL VGVMLT(NCOLB,N,JSA,A,B,C)

NCOLB	Number of columns in B.
N	Number of elements in A, C, and the number of rows in B.
JSA	Skip parameter for A. See the description of the arguments for INPROD, above.
A	Vector of length N.
B	General matrix.
C	Output vector: must be distinct from A.

8. TINV

This outline calculates the inverse of a triangular matrix. The method is based on the following identity, which can easily be verfied by the reader:

$$\begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix}^{-1} = \begin{bmatrix} U_{11}^{-1} & -U_{11}^{-1} u_{12} U_{22}^{-1} \\ 0 & U_{22}^{-1} \end{bmatrix}$$

U_{11} and U_{12} are square submatrices from the diagonal of the upper triangular matrix U; U_{12} is the remaining rectangular matrix. To invert the n x n upper triangular matrix A, we let $U = A_j$, where A_j is the submatrix of A consisting of the first j rows and the first j columns, for $j = 2,\ldots,n$. Then at each step, U_{11}^{-1} is the matrix left by the last step; U_{22} is taken as the single element A_{jj}, and U_{12} is a column vector of length j-1. The new column of U is calculated by the matrix multiplication indicated in the identity above. The process is started off by setting A_{11} equal to its reciprocal and then is continued for j=2,...,n, leaving A^{-1} in place of A.

Calling sequence: CALL TINV(N,A)

N	Number of rows and columns in A
A	Matrix to be inverted. Lower triangle is not disturbed.

9. YFACT

This routine calculates the upper triangular matrix S such that A = SS', where A is a positive definite symmetric matrix stored as a lower triangle. The method is usually attributed to Choleski and is described in Faddeeva (1), pp. 81-85, to which the reader is referred for a derivation of the method. It can be summarized as follows:

$$S_{ii} = \left(A_{ii} - \sum_{k=1}^{i-1} S_{ki}^2\right)^{1/2}$$

$$S_{ij} = \frac{A_{ij} - \sum_{k=1}^{i-1} S_{ki} S_{kj}}{S_{ii}} \qquad \text{for } j > i.$$

This routine is used in the calculation of the inverse of a symmetric matrix and is also used to calculate the matrix which is used to orthonormalze a data matrix.

Calling sequence: CALL YFACT(N,A,S)

N	Number of rows and columns in A and S.
A	Symmetric input matrix, stored as a lower triangle.
S	Upper triangular output matrix.

A and S may interlace each other as described under T2YMLT. The input matrix A must be positive definite; if it is not, YFACT will attempt to take the square root of a negative number, a fatal error in most systems.

10. YINV

This routine inverts a symmetric positive definite matrix which is stored as a lower triangle. From the input matrix A, the upper triangular matrix S such that A = SS' is calculated by calling YFACT. Then S is inverted in place by calling TINV. Finally A^{-1} is formed as $(S^{-1})'S^{-1}$, using T2YMLT. The reader should verify that the result is truly A^{-1}.

Calling sequence: CALL YINV(N,A,S)

N	Number of rows and columns in A and S.
A	Symmetric matrix stored as a lower triangle to be inverted in place.
S	Upper triangular matrix in which the intermediate matrix is stored. S may interlace A.

A must be positive definite, since YFACT is used.

11. YXPND

This routine copies the lower triangle of a symmetric matrix into the upper triangle of the same matrix, thus forming a general matrix. It is used in preparation for matrix multiplication and other operations for which special routines for symmetric matrices are not provided.

Calling sequence: CALL YXPND(N,A)

N	Number of rows and columns in A.
A	Matrix which is expanded.

Date Due

DEC 2 4

JUL 20 '76

SEP 08 '76

JAN 12 '77

AUG 04 '78

JUN 1 6 '79

JUL 17 1980

AUG 2 3 1985

Lib-26-67

MIT LIBRARIES
3 9080 003 958 771
MIT LIBRARIES
3 9080 003 927 651
MIT LIBRARIES
3 9080 003 958 664
MIT LIBRARIES
3 9080 003 958 714
MIT LIBRARIES
3 9080 003 958 623
MIT LIBRARIES
3 9080 003 958 763
MIT LIBRARIES
3 9080 003 927 693
MIT LIBRARIES
3 9080 003 958 839
MIT LIBRARIES DUPL
3 9080 003 927 669
MIT LIBRARIES
3 9080 003 927 677